Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 6 (2011) 28–33

Procedia Computer Science

Complex Adaptive Systems, Volume 1
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2011- Chicago, IL

Market-Based Solution to the Allocation of Tasks to Agents

Elad Kivelevitch[a,1,*], Kelly Cohen[a], Manish Kumar[b]

[a] *School of Aerospace Systems, University of Cincinnati, Cincinnati, OH.*
[b] *School of Dynamic Systems, University of Cincinnati, Cincinnati, OH.*

Abstract

Tasks allocation is a fundamental problem in multiagent systems. We formulate the problem as a multiple traveling salesmen problem (MTSP), which is an extension to the well known traveling salesman problem (TSP), both considered to be NP-hard combinatorial optimization problems. We propose a solution in which agents interact in an economic market to win tasks situated in an environment. The agents strive to minimize required costs, defined as either the total distance traveled by all agents or the maximum distance traveled by any agent. Using a set of simple market operations, the agents come up with a solution for task allocation. In this work we examine the processing speed of the market-based solution (MBS), as well as the quality vs. optimal solutions achieved using enumeration for a 3 agents by 8 tasks scenario. We show that the MBS is both quick and close to optimal. We then show that the MBS can be scaled to more complicated problems, by comparing its results with results from genetic algorithm (GA) and clustering. We also show the robustness of the MBS to changes in the scenario, e.g. the addition and removal of tasks or agents.

Keywords: Multiple Traveling Salesman Problem (TSP), Multi-agent systems, Market-based techniques

1. Introduction

We consider the problem of allocating a group of mobile autonomous agents to perform a set of tasks defined by location, priority and time constraints. The problem of allocating resources to tasks is a fundamental problem in optimization of a variety of autonomous systems. Some examples include allocating computer resources[1–3], allocating network resources to consumers [1, 4], allocating parts and processing machines in a factory[5], and allocating targets to unmanned vehicles[6–9]. Therefore, the benefits of a good resource allocation algorithm are far reaching.

The problem considered in this paper is defined in the following way: assign m agents to perform n tasks, so that the assignment minimizes a certain cost function, as defined in Eqs. 3-5. Each task is defined by its location, its type and a time varying benefit function. A task type is, essentially, the type of payload required to perform it. Time varying benefit functions allow the definition of task priorities and time critical tasks. Each agent is defined by its initial location, the types of tasks it can perform, and its constant speed.

*Corresponding author
Email address: kiveleeh@ucmail.uc.edu (Elad Kivelevitch)

doi:10.1016/j.procs.2011.08.008

Following the problem definition in [10], let $G = (V, A)$ be a graph, where $V = \{v_1, v_2, ..., v_n\}$ is a set of n vertices and A is a set of edges connecting these vertices. Let the benefit associated with vertex v_i be b_i (available only for one agent) and let the cost of traveling from vertex v_i to vertex v_j be d_{ij}, both assumed to be nonnegative. The cost d_{ij} is assumed to be symmetric, i.e. $d_{ij} = d_{ji}$. Furthermore, the costs are assumed to satisfy the triangle inequality, namely $d_{ij} + d_{jl} \geq d_{il}$. These assumptions are commonly used in any TSP. By defining the problem this way, there are two goals: minimizing the cost and maximizing the benefit. The best solution may be achieved in this case by visiting only vertices, whose assigned benefit is greater than the additional cost incurred by visiting them, i.e. the profit of visiting these vertices, defined as $p_i = b_i - (d_{ji} + d_{il} - d_{jl})$, is positive. This is a generalization of the TSP, because now a vertex can be visited either once or never.

In the MTSP there are m agents. Let the set of all possible routes by any agent be Ω. The MTSP is then a problem of assigning route $R_k \in \Omega$, whose associated cost is denoted by J_k and associated benefit is B_k, to agent $k \in \{1, 2, ..., m\}$. The cost and benefit of route R_k can be calculated using the definitions of costs and benefit given above as:

$$J_k = \sum_{d_{ij} \in A} a_{ik} a_{jk} d_{ij} \tag{1}$$

$$B_k = \sum_{v_i \in V} a_{ik} b_i \tag{2}$$

Where $a_{rk} \in \{0, 1\}$ are binary variables that indicate if route R_k visits vertex $v_r \in V$.

There are three possible goals in such an assignment:

- Case 1, denoted $MinSum$: Minimizing the sum of traveling distances of the entire group [10].
- Case 2, denoted $MinMax$: Minimizing the longest distance of any agent in the group [11].
- Case 3, denoted $MaxPro$: Maximizing the profit accumulated by all the agents in the group [10].

Other cost functions can be linear combinations of the ones above. These cost functions can be written as:

$$J^\star = \min \sum_{R_k \in \Omega} J_k x_k \tag{3}$$

$$J^\star = \min J_{k^+}, \tag{4}$$

$$where k^+ = argmax_{R_k \in \Omega} J_k x_k$$

$$J^\star = \min \sum_{R_k \in \Omega} (J_k - B_k) x_k \tag{5}$$

Where $x_r \in \{0, 1\}$ are binary variables that indicate whether route $R_r \in \Omega$ is part of the solution. Equations 3 and 4 are essentially the same as the cost functions defined by Shima et al [11]. The optimization process, for any of the above cases, is subject to the following constraints:

$$\sum_{R_k \in \Omega} a_{ik} x_k \leq 1 \tag{6}$$

$$\sum_{R_k \in \Omega} x_k \leq m \tag{7}$$

Equation 6 is the constraint that each vertex is visited at most once. Equation 7 is the constraint that the number of tours does not exceed the number available agents.

This problem is known to be complex, and for m agents and n tasks, assuming that each task is assigned to only one agent, we get the number of all possible assignments is given by [11]: $N_a = m^n \cdot n!$ and shows that direct enumeration of all possible assignments is intractable in problems that have more than very few agents and tasks.

The requirements from a good resource allocation algorithm are: *optimality*, *calculation time*, *scalability* and *robustness*. Optimality means that the solution reaches a cost value which is close to the optimal cost. Calculation time means that the solution has to be calculated quickly enough to be relevant, while scalability means that the problem can be extended to larger numbers of agents and tasks while the calculation time and optimality are still

acceptable. Finally, robustness means that the algorithm can overcome changes in the scenario, like addition and removal of agents or tasks.

The main contributions of this research are that we show that the market-based solution (MBS) is fast, near-optimal, robust to changes in the scenario, and outperforms other sub-optimal methods. This document is organized as follows: in section 2 the MBS is discussed in detail, followed by results in section 3 and conclusions.

2. Market-Based Solution

The solution to the problem presented in 1 is based on a market in which there are three entities: the tasks, the agents and the group administrator. The tasks are a simplified version of an agent and encapsulate the task properties and a simple behavior, which is to choose the lowest bid. Following is a description of the agents and administrator.

2.1. Agent Rules

In this work, the agent is represented by its properties, e.g. initial location, speed, and list of tasks to which it is assigned, and its behavior. The agent behavior is comprised of several basic operations: buying tasks, relinquishing tasks, taking tasks from other agents, and taking over all the assigned tasks of another agent. All the agents participate in an iterative market process in which, at each iteration, the agent buys new tasks, takes over tasks from other agents, and relinquishes tasks. Following is a short description of each basic operation.

2.1.1. Buying Tasks

Available tasks are announced at every iteration of the market process. Then, each agent announces its bid for a task, which is based on the difference in its cost function with and without the task. To decide this difference, the agent checks the order in which the task will be performed relative to the existing order of the tasks the agent is already committed to perform. The bid is made based on the minimum additional cost of adding this task. Once all the bids are announced, the task assigns itself to the lowest bidder and the agent updates its list of assigned tasks.

2.1.2. Task Takeover

Agents randomly consider the tasks of other agents. Denote agent A as the agent considering the tasks assigned to another agent, B. When agent A considers the tasks of agent B, it queries the ordered set of tasks that B is assigned to do. This allows A to consider the cost associated with each of B's tasks, and A can consider whether it can perform any of these tasks at a lower cost. If so, agent A takes this task over from agent B. This process resembles a trade in the free market, and is a greedy process looking to exploit cost reduction for the whole group.

2.1.3. Complete Takeover

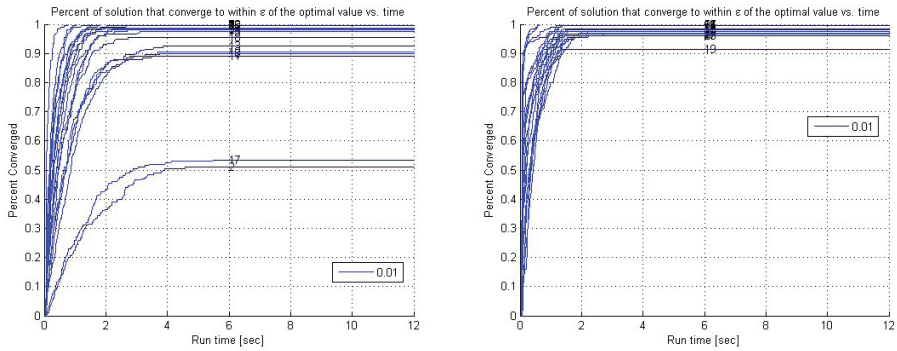
Agents are allowed to randomly take over all the tasks of another agent. Note that this process will not necessarily lead to a lower cost for the entire group, and obviously will incur greater cost to the agent that takes over the tasks. However, such behavior appears in economic markets, when one company takes over another company, at some cost to itself, but with a view toward future profits and eliminating competition. As such, the point of this process is to stir the group's cost from a local minimum and allow it to settle to a possibly lower minimum. At the end of this step, the current performance of the group as a whole is examined by a group administrator (see 2.2).

2.1.4. Relinquishing Tasks

At the end of an iteration, each agent reorders its list of tasks, by invoking any quick approximate TSP algorithm, we use convex hull [12]. Following this process, the agent recalculates the cost associated with each assigned task, and chooses to relinquish some of the tasks assigned to it. Agents randomly choose tasks to relinquish, with higher probability assigned to the tasks with higher incurred costs. The relinquished tasks from all the agents are accumulated and serve as the basis for the next iteration.

2.2. Group Administrator

The group administrator keeps track of the performance of the whole team. The group administrator makes only one decision: whether the result of an iteration is better than the result of the best iteration so far, and if so, it saves the new list of assigned tasks for each agent. This is the only group-wide function in the group, and it only deals with a higher level aspect of the solution, leaving the agents to deal with lower level decisions and calculations. This function is similar to the function that keeps the best chromosome of a genetic algorithm solution.



(a) *MinMax* (b) *MinSum*

Figure 1: Results of 20 3x8 scenarios, 200 runs each. Percent of the runs that came within $\epsilon = 1\%$ of the optimal cost

3. Results

This research is an ongoing effort and initial results of this research were presented in [13]. In this paper we present results, which summarize a research in three different directions: statistical analysis of the algorithm optimality, robustness to changes in the scenario, and scalability. All the cases were run on a single laptop, and written in MATLAB® running on a dual core 2.25GHz processor machine with 3 GB of memory.

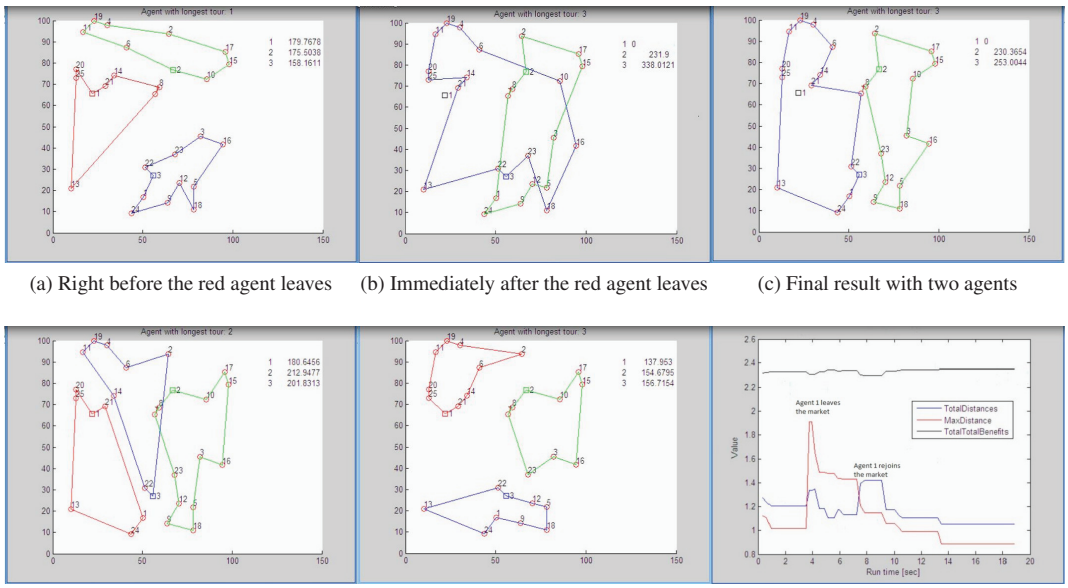
3.1. Statistical Analysis

The ability of the proposed solution to solve the problem was compared to a guaranteed optimal solution, which was computed using direct enumeration. Obviously, due to computational complexity, this solution was obtained only for a simple case, which includes 3 agents performing 8 tasks, denoted as $3x8$. Quantitative results were obtained for both the *MinSum* case (Eq. 3) and the *MinMax* case (Eq. 4). A *scenario* is defined as a given set of agent locations and task locations and a *run* is defined as one solution instance of a scenario using the MBS. The statistical analysis is based on a Monte-Carlo simulation of 200 different scenarios each with a single run, and of 20 scenarios that were run 200 runs. In [13] we showed that out of the 200 scenarios, 90% or more achieved a cost within 1% of the optimal cost for that scenario. Now we examine the percent of runs that came within $\epsilon = 1\%$ of the cost function, as a function of time, as depicted in Figure 1. It takes about 2 seconds to run for the *MinSum* case and up to 4 seconds for the *MinMax* case. For the *MinSum* case, more than 95% of the runs per each scenario getting to within 1% of the optimal cost. The *MinMax* case exhibits some variety in the results: most scenarios can be calculated quite accurately, but some have a lower percentage getting to within 1% of the optimal cost. The reason for that is, presumably, that the *MinMax* case is very susceptible to converging to local *Pareto minima*, and getting away from these local minima is harder, because they require usually more than one exchange between agents. One possible way to deal with these cases is by letting agents to relinquish tasks that are located inside another agent's polygon. However, a complete solution to Pareto optimality requires global knowledge, whereas the agents have only a local one.

3.2. Robustness

In real life, scenarios change over time. In particular, agents may become available or unavailable during the scenario, new tasks may become known and old ones may become irrelevant. Most existing solutions to the MTSP, however, do not allow these parameters to be changed during the run of the algorithm, and, if any change is made to the scenario, the algorithm should be restarted, and might never finish execution if the change rate is high.

The MBS offers a simple and intuitive way to deal with these changes, using simple rules: if an agent becomes available, let it join the bidding process; if an agent becomes unavailable, add its tasks to the list of tasks to bid on in the next iteration and exclude it from further bidding; if a new task becomes available, add it to the list of tasks; and, if a task is no longer relevant, remove it from an agent's task-list and from the market. The key here is that the MBS can accommodate these changes, while running and taking advantage of solutions that were obtained in previous steps.



(a) Right before the red agent leaves (b) Immediately after the red agent leaves (c) Final result with two agents

(d) Red agent rejoins the market (e) Final result (f) Cost variation during the run

Figure 2: Results of a scenario run with an agent leaving and rejoining the market

Figure 2 shows an example of such a scenario run. The scenario starts with 3 agents and 15 tasks. After 25 iterations, the red agent leaves the market, and the other two agents divide the tasks between them. At iteration 50, the red agent rejoins the market and bids on the tasks. The market then continues to run until a final result is found. This final result has a better cost than the result of a sub-optimal clustering solution. Thus, we show the ability of this MBS to deal with changes in the scenario.

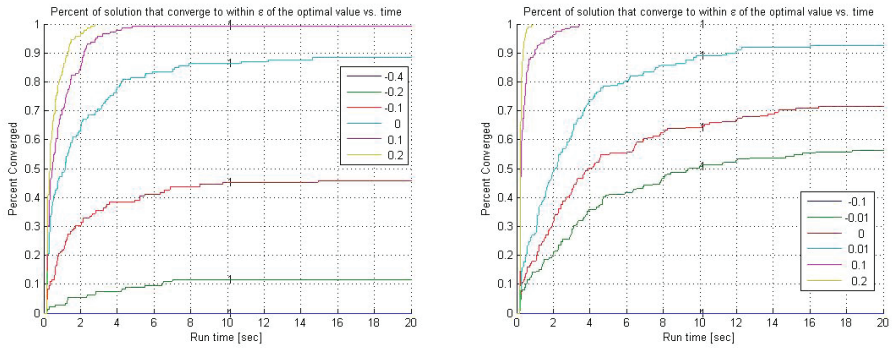
3.3. Scalability

To show scalability, we consider a problem with 5 agents and 30 tasks. For a problem of this size, direct enumeration is intractable, and therefore the optimal solution is unknown. A sub-optimal solution can be obtained by many methods, and we use two: (1) a genetic algorithm (GA) solution, based on [14], with modifications to allow fixed initial points for the agents, and (2) a clustering algorithm, which first uses k-means to cluster the tasks, and then assigns agents to clusters using direct enumeration.

A Monte-Carlo simulation is done and the results of 200 scenarios are compared. This time the optimal solution is not known, so in some cases the MBS will outperform the other two, and vice versa. In addition, having the same cost does not necessarily mean that the tasks were assigned the same way.

Figure 3 depicts the results of a 5 agents and 30 tasks case and compares the results obtained by the MBS approach with respect to the clustering algorithm. Note that the GA solution was not as efficient as the clustering solution in either case and for lack of space is not shown here. Furthermore, both the clustering and GA require more than a minute to run each solution. For the $MinMax$ case, the MBS comes within $\epsilon = 20\%$ in all cases within 2-3 seconds and within $\epsilon = 10\%$ in all cases within about 5 seconds. The MBS reaches a solution within about 15 seconds. In almost 90% of the cases it achieves the same cost ($\epsilon = 0$) as the clustering, and outperforms it by 10% ($\epsilon = -10\%$) in almost half the runs and by 20% in just above 10% of the cases.

For the $MinSum$ case the performance is generally the same in terms of time and near optimality. The MBS takes about 16 seconds to run and achieves a performance within $\epsilon = 10\%$ in all cases in about 3 seconds, but the same cost as the non-market solution is achieved only in about 70% of the runs. The graph also shows that the MBS is better than $\epsilon = 1\%$ of the non-market solution's cost in more than 90% of the runs. The MBS will outperform the non-market solution only slightly, and the difference never exceeds 10%. This is because both solutions are quite close to the optimal solution, which is easier to compute in the $MinSum$ case vs. the $MinMax$ case. More work is required to compare the MBS results to other approximate methods, and scale the MBS to even larger problems.



(a) Results of $MinMax$ vs. clustering (b) Results of $MinSum$ vs. clustering

Figure 3: Results of 200 scenarios with 5 agents and 30 tasks. Percent of the runs within ϵ of the optimal cost

4. Conclusions and Future Work

A task assignment algorithm was developed using an MBS, in which agents operate in bidding on tasks and exchanging them among themselves. The task assignment algorithm shows near optimal results in comparison to a known optimal solution, robustness to changes in the scenario, e.g. agents becoming available and unavailable, and initial results regarding the scalability were obtained by comparing the MBS to other existing sub-optimal solutions.

This is an ongoing research effort and we have many directions to extend this framework. Current work focuses on the ability of the MBS to deal with uncertainty in task locations. Another avenue of research is extending the scalability of the MBS possibly using hierarchical markets. In the longer run, we intend to add constant and time-varying priorities, to make the current MTSP formulation a multiple traveling salesmen problem with profits.

[1] R. A. Gagliano, P. A. Mitchem, Market-Based Control: a Paradigm for Distributed Resource Allocation, no. 9810222548, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 1996, Ch. Valuation of Network Computing Resources, pp. 28–52.

[2] K. Harty, D. Cheriton, Market-Based Control: a Paradigm for Distributed Resource Allocation, no. 9810222548, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 1996, Ch. A Market Approach to Operating System Memory Allocation, pp. 126–155.

[3] D. F. Ferguson, C. Nickolaou, J. Sairamesh, Y. Yemini, Market-Based Control: a Paradigm for Distributed Resource Allocation, no. 9810222548, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 1996, Ch. Economic Models for Allocating Resources in Computer Systems, pp. 156–183.

[4] K. Kuwabara, T. Ishida, Y. Nishibe, T. Suda, Market-Based Control: a Paradigm for Distributed Resource Allocation, no. 9810222548, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 1996, Ch. An Equilibratory Market-Based Approach for Distributed Resource Allocation and Its Application to Communication Network Control, pp. 53–73.

[5] A. D. Baker, Market-Based Control: a Paradigm for Distributed Resource Allocation, no. 9810222548, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 1996, Ch. Metaphor or Reality: A Case Study Where Agents Bid with Actual Costs to Schedule a Factory, pp. 184–223.

[6] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint, M. Baum, Cooperative control for autonomous air vehicles, Proceedings of the Cooperative Control Workshop, Florida.

[7] S. Rasmussen, P. Chandler, J. W. Mitchell, C. Schumacher, A. Sparks, Optimal vs. heuristic assignment of cooperative autonomous unmanned air vehicles, Proceedings of the AIAA Guidance, Navigation & Control Conference.

[8] C. Schumacher, P. Chandler, M. Pachter, Uav task assignment with timing constraints, AFRL-VA-WP-TP-2003-315United States Air Force Research Laboratory.

[9] C. Schumacher, P. Chandler, S. Rasmussen, Task allocation for wide area search munitions via network flow optimization, Proceedings of the 2001 AIAA Guidance, Navigation, and Control Conference.

[10] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problem with profits, Transportation Science 39 (2) (2005) 188 – 205. doi:10.1287/trsc.1030.0079.

[11] Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms, Computers & Operations Research 33 (11) (2006) 3252 – 3269, part Special Issue: Operations Research and Data Mining. doi:10.1016/j.cor.2005.02.039.

[12] G. Luigi, Tspconvhull, MATLAB Central (12 2008).

[13] E. Kivelevitch, K. Cohen, M. Kumar, Tasks allocation to agents using market-based techniques, in: Proceedings of the AIAA 2011 Infotech@Aerospace Conference, no. 2011-1548, AIAA, 2011.

[14] J. Kirk, Multiple variable traveling salesmen problem - genetic algorithm, Matlab Central File Exchange (June 2009).